

Komandų sistema

Intel architektūrai būdinga savybė – naudojama vienoda mnemonika visiškai skirtingoms komandoms.

d – krypties (*destination*) bitas, jis nurodo, ar operacijos kryptis yra iš procesoriaus į atmintį, ar iš atminties į procesorių;

$$d = \begin{cases} 0 - \text{iš procesoriaus registro į atmintį} \\ 1 - \text{iš atminties į procesoriaus registrą} \end{cases}$$

w - pločio (*width*) bitas, jis nurodo operacijoje dalyvaujančio lauko ilgį;

$$w = \begin{cases} 0 - \text{operandas baitas} \\ 1 - \text{operandas žodis} \end{cases}$$

mod lauko reikšmės:

$$\text{mod} = \begin{cases} 00 - \text{operandas r/m atmintyje (r/m = m), poslinkio nėra;} \\ 01 - \text{operandas r/m atmintyje (r/m = m), vieno baito poslinkis;} \\ 10 - \text{operandas r/m atmintyje (r/m = m), dviejų baitų poslinkis;} \\ 11 - \text{operandas r/m registre (r/m = r);} \end{cases}$$

bet.op. – betarpiškas operandas.

j.b	v.b
-----	-----

adr.j.b. – adreso jaunesnysis baitas

adr.v.b. – adreso vyresnysis baitas

sreg – segmento registras;

$$\text{sreg} = \begin{cases} 00 - \text{ES;} \\ 01 - \text{CS;} \\ 10 - \text{SS;} \\ 11 - \text{DS;} \end{cases}$$

portas – porto numeris diapazone 0 – 255

posl.j.b. – poslinkio jaunesnysis baitas

posl.v.b. – poslinkio vyresnysis baitas

seg.rer.j.b. – segmento registro jaunesnysis baitas

seg.reg.v.b. – segmento registro vyresnysis baitas

Bendrosios mikroprocesoriaus komandos

MOV – MOVE – persiuntimas

1. registras \leftrightarrow registras / atmintis;

1000 10dw mod reg r/m [poslinkis];

d = 0 : reg \rightarrow r / m

d = 1: r / m \rightarrow reg

2. betarpiškas operandas \rightarrow registras / atmintis;

1100 011w mod 000 r/m [poslinkis] bet.op1 [bet.op.2, jei w = 1]

Krypties bito nėra, nes kryptis jau yra apibrėžta.

Laukas reg nereikalingas, jis yra naudojamas kaip operacijos kodo išplėtimas.

Poslinkis suformuojamas priklausomai nuo lauko *mod* reikšmės.

3. betarpiškas operandas \rightarrow registras

1011 w reg bet.op.1 [bet.op.2, jei w = 1]

4. atmintis \rightarrow akumulatorius

1010 000w adr.j.b adr.j.b adr.v.b.

Komandos formatas yra toks, kad registras ir operandas betarpiškai nenurodyti.

Iš operacijos kodo yra žinoma, kad registras yra akumulatorius.

5. akumulatorius \rightarrow atmintis

1010 001w adr.j.b adr.j.b adr.v.b.

Pastaba. Atvejų 4. ir 5. negalima sujungti į vieną 101000 d w adr.j.b adr.j.b adr.v.b. , nes bitas d apibėžia priešingos krypties perdavimą, negu yra nurodyta 4. ir 5. atvejais.

6. registras / atmintis \leftrightarrow segmento registras

1000 11d0 mod 0 sreg r/m [poslinkis]

sreg negali būti lygus 01, nes negalima nusiųsti reikšmės į registrą CS, nes priešingu atveju priskyrimo veiksmu pakeistume CS registro reikšmę, o ji gali būti pakeista tik valdymo perdavimo komandomis.

Nėra w bito, nes operandas visada 2 baitų.

PUSH – įdėjimas į steką.

Prieš rašant į steką reikia sumažinti steko rodyklę : SP = SP - 2 (steke operuojame žodžiais).

1. registras / atmintis \rightarrow stekas

1111 1111 mod 110 r/m [poslinkis]

Tai yra vieno operando komanda, laukas *reg* yra naudojamas kaip operacijos kodo išplėtimas.

2. registras → stekas
0101 0reg

Nėra varianto, ar registras baitinis ar žodinis, nes vienareišmiškai jis yra žodinis.

3. segmento registras → stekas
000 sreg 110

POP – paėmimas iš steko.

Paėmus iš steko reikia padidinti steko rodyklę : $SP = SP + 2$ (steke operuojame žodžiais).

1. stekas → registras / atmintis
1000 1111 mod 110 r/m [poslinkis]

2. stekas → registras
01011 reg

3. stekas → segmento registras
000 sreg 111

sreg negali būti lygu 01, nes tada pakeistume CS registro reikšmę, o ji gali būti pakeista tik valdymo perdavimo komandomis.

XCHG – eXCHanGe – apsisikeitimas reikšmėmis.

1. registras / atmintis ↔ registras
1000 011w mod reg r/m [poslinkis]

2. registras ↔ akumulatorius
10010 reg

Nėra w bito, nes šia komanda vykdomas apsisikeitimas tarp akumulatoriaus AX ir dviejų baitų registro, $reg = \{ AX, CX, DX, BX, SP, BP, SI, DI \}$

Komanda XCHG AX, AX yra ekvivalenti NOP, t.y., operacijos kodas yra 1001 0000, t.y., 90h.

OUT – išvedimas į portą.

1. Fiksuotas portas.
1110 011w portas;

Registre AL (jei w = 0) arba registre AX (jei w = 1) esanti reikšmė nusiunčiama į portą, nurodytą komandoje porto numeriu.

2. Kintamas portas.

1110 111w

Porto numeris yra imamas iš registro DX, todėl galima nurodyti 65 536 skirtingus portus. Vykdam komandą, reikšmė iš AL (jei $w = 0$) arba iš AX (jei $w = 1$) yra siunčiama į portą.

IN – nuskaitymas iš porto.

1. Fiksuotas portas.

1110 010w portas

Vykdam komandą, iš porto su nurodytu numeriu padedama reikšmė į registrą AL (jei $w = 0$) arba į registrą AX (jei $w = 1$).

2. Kintamas portas.

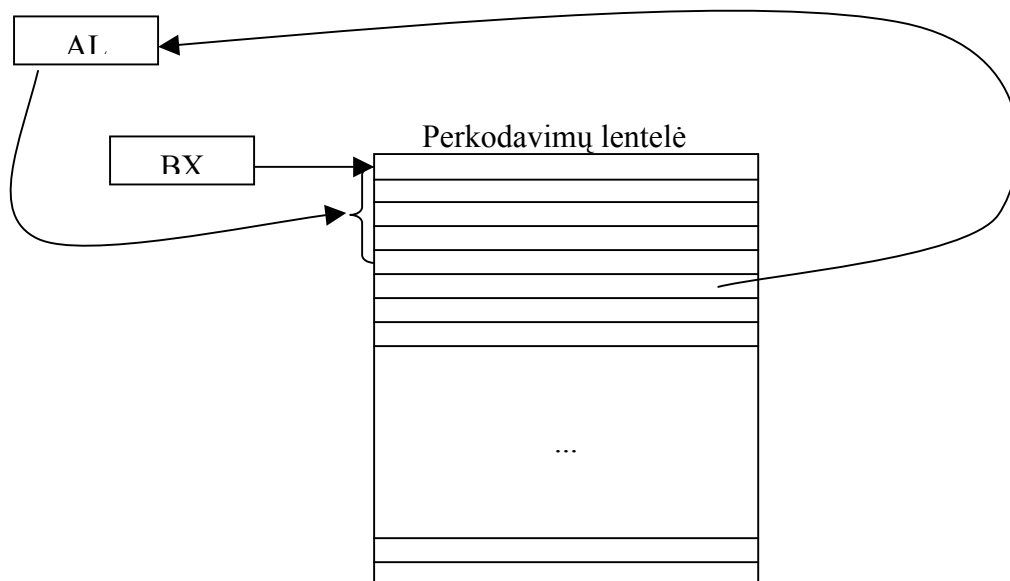
1110 110w

Porto numeris yra imamas iš registro DX, todėl galima nurodyti 65 536 skirtingus portus. Vykdam komandą, reikšmė iš porto užrašoma į registrą AL (jei $w = 0$) arba į AX (jei $w = 1$). Kai atliekamas veiksmas yra žodinis, tai tie 2 baitai yra imami ne iš dviejų gretimų portų, bet abu iš to pačio porto, skaitant iš jo per du taktus.

XLAT – transLATE – kodo paėmimas pagal poslinkį.

1101 0111

Komanda naudojama duomenų perkodavimui. Registre BX esantis adresas rodo į perkodavimo lentelę. Imamas registre AL esantis baitas ir jis traktuojamas kaip poslinkis nuo perkodavimo lentelės pradžios. Vykdam komandą, reikšmė, esanti perkodavimo lentelėje su poslinkiu, nusiunčiama į registrą AL.



LEA – Load Effective Address – pakrauti (patalpinti) vykdomąjį adresą.
1000 1101 mod reg r/m [poslinkis]

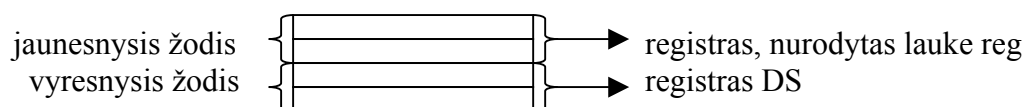
Komanda atlieka vykdomojo (efektyvaus) adreso nusiuntimą į registrą.

reg yra bet kuris 16 bitų registras, išskyrus segmento registrus.

Kadangi antras operandas būtinai yra atmintis, *mod* negali būti lygus 11.

LDS – Load pointer using DS – bendro registro ir DS užpildymas.
1100 0101 mod reg r/m [poslinkis]

Komanda atlieka atminties dvigubo žodžio jaunesniųjų adresu dviejų baitų nusiuntimą į registrą, nurodytą lauke *reg*, ir vyresnių adresų dviejų baitų nusiuntimą į registrą DS.

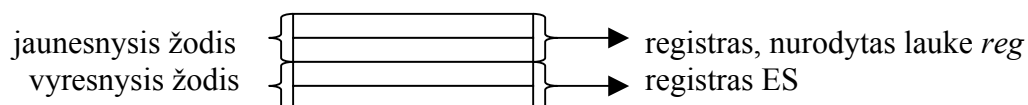


Kadangi antras operandas būtinai yra atmintis, *mod* negali būti lygus 11.

Komandos lauku *r/m* [poslinkis] nurodomas dvigubo žodžio lauką atmintyje; operando adresas suformuojamas panaudojant registrą DS : DS + *efektyvus adresas* .

LES – Load pointer using ES – bendro registro ir ES užpildymas.
1100 0100 mod reg r/m [poslinkis]

Komanda LES yra panaši į komandą LDS: atlieka atminties dvigubo žodžio jaunesniųjų adresu dviejų baitų nusiuntimą į registrą, nurodytą lauke *reg*, ir vyresnių adresų dviejų baitų nusiuntimą į registrą ES.



Kadangi antras operandas būtinai yra atmintis, *mod* negali būti lygus 11.

Komandos lauku *r/m* [poslinkis] nurodomas dvigubo žodžio lauką atmintyje; operando adresas suformuojamas panaudojant registrą DS (ne registrą ES !) : DS + *efektyvus adresas* .

LAHF – Load AH status Flag – registro AH užpildymas registro SF jaunesniu baidu.
1001 1111

SAHF – Save AH status Flag – registro AH nusiuntimas į registro SF jaunesnįjį baitą.

1001 1110

Pastaba: Intel 8080 procesoriuje buvo komandos LAHF ir SAHF. Į Intel 8088 komandų sistemą jos įtrauktos tam, kad veiktų programos, kurios buvo rašytos senesniame, Intel 8080, procesoriui.

PUSHF – PUSH status Flag – registro SF nusiuntimas į steką.

1001 1100

POPF – POP status Flag – registro SF užpildymas iš steko viršūnės.

1001 1101

Tai yra vienintelė komanda, kuri pakeičia visą registre SF esančią reikšmę iš karto. Kitomis komandomis galima keisti atskirus bitukus.

ADD – sudėtis

1. registras + registras / atmintis

0000 00dw mod reg r/m [poslinkis]

2. registras / atmintis + betarpiškas operandas

1000 00sw mod 000 r/m [poslinkis] bet.op.j.b. [bet.op.v.b]

Jeigu s:w = 01, tada betarpiškas operandas yra imamas iš 16 komandos bitų.

Jeigu s:w = 11, tada betarpiškas operandas yra imamas iš 8 komandos bitų ir išplečiamas iki 16 bitų pagal ženklo plėtimo taisyklę: jeigu vyriausiam bite buvo 1, tada išplečiame vienetais, o jeigu vyriausiam bite buvo 0, tada išplečiame nuliais.

Jeigu w = 0, tai s gali būti lygu 0 arba 1 – betarpiškas operandas bus vieno baito.

3. akumuliatorius + betarpiškas operandas

0000 010w bet.op.j.b [bet.op.v.b., jei w = 1]

ADC – ADd with Carry – sudėtis, įskaitant pernešimo bitą CF.

Komanda yra analogiška komandai ADD, tik papildomai pridedamas CF:

operandas1 + operandas2 + CF

1. registras + registras / atmintis

0001 00dw mod reg r/m [poslinkis]

2. registras / atmintis + betarpiškas operandas

1000 00sw mod 010 r/m [poslinkis] bet.op.j.b. [bet.op.v.b]

Jeigu s:w = 01, tada betarpiškas operandas yra imamas iš 16 komandos bitų.

Jeigu $s:w = 11$, tada betarpiškas operandas yra imamas iš 8 komandos bitų ir išplečiamas iki 16 bitų pagal ženklų plėtimo taisyklę: jeigu vyriausiam bite buvo 1, tada išplečiame vienetais, o jeigu vyriausiam bite buvo 0, tada išplečiame nuliais.

Jeigu $w = 0$, tai s gali būti lygu 0 arba 1 – betarpiškas operandas bus vieno baido.

3. akumulatorius + betarpiškas operandas
0001 010w bet.op.j.b [bet.op.v.b., jei $w = 1$]

INC – INCrement – didinimas vienetu.

1. registras / atmintis

1111 111w mod 000 r/m [poslinkis]

2. registras

01000 reg

Šia komanda inkrementuojami tik žodiniai registrai.

SUB – SUBtract – atimtis.

1. registras / atmintis ~ registras

Abu operandai gali būti pirmu arba antru operandu. Rezultatas užrašomas į pirmąjį.

0010 10dw mod reg r/m [poslinkis]

Nors atimtį realizuoja sudėtis su priešingu ženklu, bet interpretuojamo lygio komandų kodai turi skirtis, tam kad komanda sudarytų antrą operandą papildomu kodu, ir tada pritaikytų aparatūrinį sudėties veiksmą.

2. registras / atmintis - betarpiškas operandas

1000 00sw mod 101 r/m [poslinkis] bet.op.j.b. [bet.op.v.b.]

Jeigu $s:w = 01$, tada betarpiškas operandas yra imamas iš 16 komandos bitų.

Jeigu $s:w = 11$, tada betarpiškas operandas yra imamas iš 8 komandos bitų ir išplečiamas iki 16 bitų pagal ženklų plėtimo taisyklę: jeigu vyriausiam bite buvo 1, tada išplečiame vienetais, o jeigu vyriausiam bite buvo 0, tada išplečiame nuliais.

Jeigu $w = 0$, tai s gali būti lygu 0 arba 1 – betarpiškas operandas bus vieno baido.

3. akumulatorius - betarpiškas operandas

0010 110w bet.op.j.b. [bet.op.v.b., $w = 1$]

SBB – SuBtraction with Borrow – atimtis su pasiskolinimu

Komanda yra analogiška komandai SUB, tik papildomai atimamas CF:

operandas1 - operandas2 – CF

1. registras / atmintis ~ registras

Abu operandai gali būti pirmu arba antru operandu. Rezultatas užrašomas į pirmąjį.

0001 10dw mod reg r/m [poslinkis]

2. registras / atmintis - betarpiškas operandas

1000 00sw mod 011 r/m [poslinkis] bet.op.j.b. [bet.op.v.b]

Jeigu s:w = 01, tada betarpiškas operandas yra imamas iš 16 komandos bitų.

Jeigu s:w = 11, tada betarpiškas operandas yra imamas iš 8 komandos bitų ir išplečiamas iki 16 bitų pagal ženklo plėtimo taisyklę: jeigu vyriausiame bite buvo 1, tada išplečiame vienetais, o jeigu vyriausiame bite buvo 0, tada išplečiame nuliais.

Jeigu w = 0, tai s gali būti lygu 0 arba 1 – betarpiškas operandas bus vieno baito.

3. akumulatorius - betarpiškas operandas

0001 110w bet.op.j.b. [bet.op.v.b., w = 1]

DEC – DECrement – sumažinimas vienetu.

1. registras / atmintis

1111 111w mod 001 r/m [poslinkis]

2. registras

01001 reg

Šia komanda dekrementuojami tik žodiniai registrai.

CMP – CoMPare – palyginimas.

Komanda naudojama požymių formavimui, veikia panašiai kaip ir komanda SUB, tačiau neįsimena rezultato.

Požymių formavimas:

Jeigu operandai yra skaičiai be ženklo:

	OF	SF	ZF	CF
op.1 > op.2	N	N	0	0
op.1 = op.2	N	N	1	0
op.1 < op.2	N	N	0	1

op. – operandas;

N – nesvarbu.

Jeigu operandai yra skaičiai su ženklu:

	OF	SF	ZF	CF
op.1 > op.2	0 / 1	0 / 1	0	N

op.1 = op.2	0	0	1	N
op.1 < op.2	0 / 1	0 / 1	0	N

op. – operandas;

N – nesvarbu.

0 / 1 – 0 arba 1, priklausomai, nuo konkrečių lyginamų reikšmių.

Skaičių su ženklu palyginimas, diapazone -128 – 127 :

	SF	OF
$0 < A - B \leq 127$	0	0
$A - B > 127$	1	1
$-128 \leq A - B < 0$	1	0
$A - B < -128$	0	1

Taigi $A > B$, kai : $\left. \begin{array}{l} 1. OF = 0; SF = 0 \\ 2. OF = 1; SF = 1 \end{array} \right\}$,t.y., kai OF lygu SF

$A < B$, kai : $\left. \begin{array}{l} 1. OF = 1; SF = 0 \\ 2. OF = 0; SF = 1 \end{array} \right\}$,t.y., kai OF nelygu SF

Algoritmas požymiui OF nustatyti:

Vyriausiąją baido bitą pažymėkimė s , taip pat stebėsimė bitą, einantį po bito s (vyresnį už bitą s), jį pažymėkimė x . Skirkime du atvejus:

1. *I operandas* > *II operandas*, t.y., rezultatas teigiamas:

jeigu $x = 0$ ir $s = 0$, tada $OF = 0$;

jeigu $x = 0$ ir $s = 1$, tada $OF = 1$.

2. *I operandas* < *II operandas*, t.y., rezultatas neigiamas:

jeigu $x = 1$ ir $s = 1$, tada $OF = 0$;

jeigu $x = 1$ ir $s = 0$, tada $OF = 1$.

Pavyzdžiai:

1. 10 - 5

$(10)_{10} = (1010)_2$

$(5)_{10} = (101)_2$,

$$\begin{array}{r} 0000\ 0101 \\ 1111\ 1010 \\ + \quad \quad 1 \\ \hline 1111\ 1011 \end{array} = (-5)_{10}$$

$$\begin{array}{r} 0000\ 1010 \\ + 1111\ 1011 \\ \hline 10000\ 0101 \end{array} \quad OF = 0, SF = 0$$

2. $100 - (-101)$

$(100)_{10} = (1100100)_2$

$(101)_{10} = (1100101)_2$

$$\begin{array}{r} 0110\ 0100 \\ + \underline{0110\ 0101} \\ 1100\ 1001 \end{array} \quad \text{OF} = 1, \text{SF} = 1$$

3. $-3 - (7)$

$(3)_{10} = (11)_2$,

$$\begin{array}{r} 0000\ 0011 \\ 1111\ 1100 \\ + \underline{1} \\ 1111\ 1101 \end{array} = (-3)_{10}$$

$(7)_{10} = (111)_2$,

$$\begin{array}{r} 0000\ 0111 \\ 1111\ 1000 \\ + \underline{1} \\ 1111\ 1001 \end{array} = (-7)_{10}$$

$$\begin{array}{r} 1111\ 1101 \\ + \underline{1111\ 1001} \\ 11111\ 0110 \end{array} \quad \text{OF} = 0, \text{SF} = 1$$

4. $-101 - (100)$

$(101)_{10} = (1100101)_2$

$$\begin{array}{r} 0110\ 0101 \\ 1001\ 1010 \\ + \underline{1} \\ 1001\ 1011 \end{array} = (-101)_{10}$$

$(100)_{10} = (110010)_2$

$$\begin{array}{r} 0110\ 0100 \\ 1001\ 1011 \\ + \underline{1} \\ 1001\ 1100 \end{array} = (-100)_{10}$$

$$\begin{array}{r} 1001\ 1011 \\ + \underline{1001\ 1100} \\ 10011\ 0111 \end{array} \quad \text{OF} = 1, \text{SF} = 0$$

1. registras / atmintis ~ registras

Abu operandai gali būti pirmu arba antru operandu.

0011 10dw mod reg r/m [poslinkis]

2. registras / atmintis - betarpiškas operandas

1000 00sw mod 111 r/m [poslinkis] bet.op.j.b. [bet.op.v.b.]

Jeigu $s:w = 01$, tada betarpiškas operandas yra imamas iš 16 komandos bitų.

Jeigu $s:w = 11$, tada betarpiškas operandas yra imamas iš 8 komandos bitų ir išplečiamas iki 16 bitų pagal ženklų plėtimo taisyklę: jeigu vyriausiam bite buvo 1, tada išplečiame vienetais, o jeigu vyriausiam bite buvo 0, tada išplečiame nuliais.

Jeigu $w = 0$, tai s gali būti lygu 0 arba 1 – betarpiškas operandas bus vieno baido.

3. akumulatorius - betarpiškas operandas

0011 110w bet.op.j.b. [bet.op.v.b., $w = 1$]

MUL – MULtiplj – skaičių be ženklų daugyba

1111 011w mod 100 r/m [poslinkis]

Jeigu dauginami baitai, tai:

registre AL esanti reikšmė · operandas 2 → registras AX

Jeigu dauginami žodžiai, tai:

registre AX esanti reikšmė · operandas 2 → registrai [DX, AX]

Jeigu daugybos rezultato vyresnioji pusė yra nulinė, tada $CF = 0$ ir $OF = 0$, priešingu atveju $CF = 1$ ir $OF = 1$.

IMUL – Integer MULtiplj – skaičių su ženklų daugyba

1111 011w mod 101 r/m [poslinkis]

Jeigu dauginami baitai, tai:

registre AL esanti reikšmė · operandas 2 → registras AX

Jeigu dauginami žodžiai, tai:

registre AX esanti reikšmė · operandas 2 → registrai [DX, AX]

Jeigu daugybos rezultato vyresniosios pusės kiekvieno bito reikšmė lygi jaunesniosios pusės vyriausiam bitui, t.y., daugybos rezultato vyresnioji pusė yra tik jaunesniosios pusės ženklų išplėtimas, tada rezultatas tilpo jaunesniojoje pusėje, todėl $CF = 0$ bei $OF = 0$. Priešingu atveju, $CF = 1$ ir $OF = 1$.

DIV – DIVide – skaičių be ženklų dalyba.

1111 011w mod 110 r/m [poslinkis]

Jeigu $w = 0$, tada daliklis yra baitas, ir:

*registre AX esanti reikšmė : operandas 2 → dalmuo registre AL
→ liekana registre AH*

Jeigu $w = 1$, tada daliklis yra žodis, ir:

*registruose [DX, AX] esanti reikšmė : operandas 2 → dalmuo registre AX
→ liekana registre DX*

Jeigu dalmuo netelpa registre AL (kai $w = 0$) arba AX (kai $w = 1$), tada generuojamas pertraukimas – dalyba iš nulio.

IDIV – Integer DIVide – skaičių su ženklu dalyba.

1111 011w mod 111 r/m [poslinkis]

Jeigu $w = 0$, tada daliklis yra baitas, ir:

*registre AX esanti reikšmė : operandas 2 → dalmuo registre AL
→ liekana registre AH*

Jeigu $w = 1$, tada daliklis yra žodis, ir:

*registruose [DX, AX] esanti reikšmė : operandas 2 → dalmuo registre AX
→ liekana registre DX*

Dalybos iš nulio pertraukimas kyla tada, kai dalmuo-baitas netelpa diapazone: -128 – 127, arba dalmuo-žodis netelpa diapazone: -32768 – 32767.

NEG – NEGative – ženklo keitimas

1111 011w mod 011 r/m [poslinkis]

Rezultatas yra analogiškas rezultatui, kurį gautume atlikę tokį veiksmą:

0 - operandas → operandas

Faktiškai, pritaikius komandą NEG, operandą gauname papildomu kodu.

Veiksmai su dešimtainio formato skaičiais

Tai yra veiksmų, atliktų bendrosiomis komandomis patikslinimas, kad rezultatas būtų korektiškas.

Pastaba. Požymis AF fiksuoja pernešimą iš jaunesnio pusbaičio, o požymis CF fiksuoja pernešimą iš baito.

AAA – Ascii Adjust for Addition

0011 0111

ASCII tai yra simbolinis kodavimas, kuriam yra ekvivalentūs dešimtainio nesupakuoto formato skaičiai, nes vienas toks skaičius užima vieną baitą.

Komanda yra koreguojama reikšmė, esanti registre AL, kuri pertvarkoma į teisingą dešimtainį nesupakuotą skaičių, kurios vyresnis pusbaitis yra 0h, o jaunesniajame pusbaityje yra skaičiai iš diapazono 0h – 9h.

Jeigu korekcijos rezultatas yra didesnis už 9h, nėra kaip to užfiksuoti dešimtainiu nesupakuotu skaičiumi, todėl komanda AAA suformuoja požymį CF = 1 ir registre AH esančią reikšmę padidina vienetu.

Komandos AAA veikimas priklauso nuo registro SF požymio AF. Veikiant komandai AAA, požymis AF yra „numetamas”, nes baigus vykdyti komandą, į šitą požymį jau yra sureaguota.

Komandos AAA veikimo algoritmas:

```
if ( (AL and 0Fh) > 9 or (AF = 1) ) then
    AL := AL + 6
    AH := AH + 1
    AF := 1
    CF := 1
else
    AF := 0
    CF := 0
endif
AL := AL and 0Fh
```

Pavyzdžiai:

1. $3 + 7 = 0A \rightarrow 10 \rightarrow 00$

Komandos AAA veikimo rezultate, skaičiaus A reikšmė yra padidinama šešiais dėl pagrindų skirtumo: sudedant pozicijos svoris yra 16, o norime gauti reikšmę, kurios pozicijos svoris yra 10.

Tačiau teisingas dešimtainis nesupakuotas skaičius vyresniame pusbaityje turi turėti nulį, todėl rezultatas yra dar koreguojamas iki 00. Gautas rezultatas buvo didesnis už 9, todėl nustatomas požymis CF = 1, ir registre AH esanti reikšmė padidinama vienetu.

2. $8 + 9 = 11 \rightarrow 17 \rightarrow 07$

1-ame pavyzdyje koregavome rezultatą 0A, požymis AF buvo lygus 0. Tarkime, kad po sudėties gavome 11, tada papildomo pernešimo požymis AF nustatomas lygus 1.

Skaičių 11 padidiname šešiais ir gauname 17. Kadangi vyresniame pusbaityje turi būti 0, tai ši reikšmė pakoreguojama į 07. Nustatomas požymis CF = 1, registre AH esanti reikšmė padidinama vienetu, „numetamas” požymis AF.

DAA – Decimal Adjust for Addition

0010 0111

Komanda yra koreguojama reikšmė, esanti registre AL, kuri pertvarkoma į teisingą dešimtainį supakuotą skaičių diapazone. Jeigu po korekcijos rezultatas viršija 99, tai nustatomas požymis CF = 1

Komandos AAA veikimo algoritmas:

```
if ( (AL and 0Fh) > 9 or (AF = 1) ) then
    AL := AL + 6
    AF := 1
endif
```

```

if ( (AL > 9Fh) or (CF = 1) ) then
    AL := AL + 60h
    CF := 1
endif

```

Pavyzdžiai:

1. $24 + 27 = 4B \rightarrow 51$

Sudėties rezultatą 4B gauname naudodami esamą sudėties komandą. Komanda DAA prideda 6, ir turime korektišką dešimtainį supakuotą skaičių-rezultatą: 51.

2. $29 + 28 = 51 \rightarrow 57$

51 būtų teisingas dešimtainis supakuotas skaičius, bet atliekant sudėtį buvo fiksuotas pernešimas į vyresnį pusbaitį: požymis AF = 1, todėl yra pritaikoma komanda DAA.

3. $62 + 64 = C6 \rightarrow 26$

Koreguojant sudėties rezultatą C6, gauname 126, o tai yra daugiau už 99 ir viršija dešimtainių supakuotų skaičių diapazoną. Todėl yra nustatomas požymis CF = 1.

4. $66 + 65 = CB \rightarrow D1 \rightarrow 31$

Koreguojant sudėties rezultatą CB, gauname D1, o tai irgi yra neteisingas dešimtainis skaičius, todėl komanda DAA koreguoja dar kartą, taip gaunamas 131, o tai yra daugiau už 99 ir viršija dešimtainių supakuotų skaičių diapazoną. Todėl yra nustatomas požymis CF = 1.

5. $46 + 55 = 9B \rightarrow A1 \rightarrow 01$

Koreguojant sudėties rezultatą 9B, gauname A1, o tai yra vis dar neteisingas dešimtainis skaičius, todėl komanda DAA koreguoja dar kartą, taip gaunamas 101, o tai yra daugiau už 99 ir viršija dešimtainių supakuotų skaičių diapazoną. Todėl yra nustatomas požymis CF = 1.

AAS – Ascii Adjust for Substraction

0011 1111

Komanda yra koreguojama reikšmė, esanti registre AL, kuri pertvarkoma į teisingą dešimtainį nesupakuotą skaičių, kurios vyresnis pusbaitis yra 0h, o jaunesniajame pusbaityje yra skaičiai iš diapazono 0h – 9h. Jeigu korekcijos rezultatas yra daugiau už 9, tai komanda AAS registro AH reikšmę sumažina vienetu ir nustato požymį CF = 1, priešingu atveju, CF = 0.

Atimties, kurios rezultatą koreguojame komanda AAS, operandai yra iš diapazono 0 – 9, todėl po atimties turime :

1. rezultatas ≥ 0 : $0 \leq \text{rezultatas} \leq 9$

Korekcija nereikalinga.

2. rezultatas < 0 : $-9 \leq \text{rezultatas} < 0$

$-9 = F7 \leq \text{rezultatas} \leq FF = -1$

$F7 \leq \text{rezultatas} \leq FF$

 -6

$1 \leq \text{rezultatas} \leq 9$, AH sumažinama vienetu.

Komandos AAS veikimo algoritmas:

```
if ( (AX and 0Fh) > 9 or (AF = 1) ) then
    AL := AL - 6
    AH := AH - 1
    AF := 1
    CF := 1
else
    AF := 0
    CF := 0
endif
AL := AL and 0Fh
```

Pavyzdys: 5 - 8

$(5)_{10} = (101)_2$,

$(8)_{10} = (1000)_2$

$$\begin{array}{r} 0001\ 0000 \\ 1110\ 1111 \\ + \quad \quad 1 \\ \hline 1111\ 1000 \end{array} = (-8)_{10}$$

$$\begin{array}{r} 0000\ 0101 \\ + 1111\ 1000 \\ \hline 1111\ 1101 \end{array} = (FD)_{16}$$

$(1)5 - 8 = 7$

$FD \rightarrow F7 \rightarrow 07$

Rezultatui FD reikalinga korekcija, atimama 6. Nesupakuoto dešimtainio skaičiaus vyresiaame pusbaityje turi būti nulis, todėl F7 koreguojame ir turime rezultatą 7. Tokios korekcijos eigoje komanda AAS vienetu sumažino registre AH esančią reikšmę.

DAS – Decimal Adjust for Subtraction

0010 1111

Komanda yra koreguojama reikšmė, esanti registre AL, kuri pertvarkoma į teisingą dešimtainį supakuotą skaičių diapazone. Jeigu po korekcijos rezultatas viršija 99, tai komanda DAS nustato požymį CF = 1, priešingu atveju, CF = 0.

Atimties, kurios rezultatą koreguojame komanda DAS, operandai yra iš diapazono 0 – 99. Po atimties gauname reikšmę xy ir tikriname:

1. Ar CF = 1?
2. Ar AF = 1?
3. Ar x yra teisingas dešimtainis skaitmuo?
4. Ar y yra teisingas dešimtainis skaitmuo?

Tada elgiamasi taip:

1. Jeigu AF = 1 arba y yra neteisingas dešimtainis skaitmuo, tai $xy = xy - 6$.
2. Jeigu CF = 1 arba x yra neteisingas dešimtainis skaitmuo, tai $xy = xy - 60$.

Pavyzdys: 11 - 19

Ši kartą atlikime skaičiavimus šešioliktainėje sistemoje:

$$\begin{array}{r} 11 \\ - 19 \\ \hline F8 \\ - 6 \\ \hline F2 \\ - 60 \\ \hline 92 \end{array}$$

(1)11 - 19 = 92

F8 → F2 → 92

Skaičiuojant rezultatą F8 nustatytas požymis AF = 1, todėl jį koreguojame atimdami 6, gauname F2. F yra neteisingas dešimtainis skaičius, o ir CF nustatytas lygus vienam, todėl atimame 60, gauname 92.

Komandos DAS veikimo algoritmas:

```
if ( (AL and 0Fh) > 9 or (AF = 1) ) then
    AL := AL - 6
    AF := 1
endif
if ( (AL > 9Fh) or (CF = 1) ) then
    AL := AL - 60h
    CF := 1
endif
```

AAM – Ascii Adjust for Multiplication

1101 0100 0000 1010

Komanda yra taikoma po dviejų dešimtainių nesupakuotų skaičių baitinės daugybos, kai sandauga yra registre AX, o šiuo atveju, t.y., kai yra dauginami du dešimtainiai nesupakuoti skaičiai, rezultatas telpa registre AL (maksimalus rezultatas yra $9 \cdot 9 = 81$).

Komanda AAM padalina registre AL esančią reikšmę iš 10, tada dalmenį užrašo į registrą AH, o liekaną – į registrą AL:

```
AH := AL div 10
AL := AL mod 10
```

Taip du skaičiai yra atskiriami, ir registruose AH ir AL turime teisingus dešimtainius nesupakuotus skaičius, kai vyresniame pusbaityje yra nulis, o jaunesniame pusbaityje yra skaitmuo iš diapazono: 0h – 9h.

Pavyzdys: jeigu dalybos rezultate gauname 79, tai po komandos AAM suveikimo, registre AH yra 7, o registre AL yra 9.

Pastaba. Supakuotiems dešimtainiams skaičiams nėra daugybos korekcijos komandos.

AAD – Ascii Adjust for Division
1101 0101 0000 1010

Komanda atlieka dalybos rezultato korekciją dešimtainiams nesupakuotiems skaičiams. Ši korekcijos komanda skiriasi nuo kitų korekcijos komandų tuo, kad ji yra taikoma ne rezultatui, o paruošia duomenis dešimtainių nesupakuotų skaičių dalybai.

Dešimtainių nesupakuotų skaičių dalybos korekcijos komanda taria, kad dalmuo yra registre AX kaip dviejų baitų dešimtainis nesupakuotas skaičius, kurio vyresnis skaitmuo yra registre AH, kurio vyresnis pusbaitis yra nulis.

Komanda atlieka tokius veiksmus:

AL := AH * 10 + AL
AH := 0

Tada yra atliekama dalybos komanda DIV.

Pastaba. Supakuotiems dešimtainiams skaičiams nėra dalybos korekcijos komandos.

Konvertavimas

CBW – Convert Byte to Word
1001 1000

Ši komanda yra taikoma skaičiams su ženklu, tokiems kurie yra baito diapazone, t.y., -128 – 127. Registro AL vyriausiasis bitas yra išskleidžiamas į registrą AH. Taip reikšmė baite yra pervedama į reikšmę žodyje.

CWD – Convert Word to Double word
1001 1001

Ši komanda yra taikoma skaičiams su ženklu, tokiems, kurie yra dviejų baitų – žodžio diapazone, t.y., -32768 – 32767. Registro AX vyriausiasis bitas yra išskleidžiamas į registrą DX. Taip reikšmė baite yra pervedama į reikšmę žodyje.

Komanda gali būti naudojama, pvz., dalybos veiksmo IDIV paruošimui, nes žodyje esančių reikšmių dalyba reikalauja, kad pirmas operandas būtų keturiuose baituose, t.y., registruose [DX, AX].

Loginės komandos

NOT – loginis paneigimas.
1111 011w mod 010 r/m [poslinkis]

Vykdant komandą, lauko bitai yra invertuojami.

SHL / SAL – SHift logical Left / Shift Arithmetic Left – loginis postūmis į kairę / aritmetinis postūmis į kairę.

1101 00vw mod 100 r/m [poslinkis]

Jeigu $v = 0$, tai pastumiama į kairę per vieną poziciją.

Jeigu $v = 1$, tai postūmio skaitliukas yra registro CL reikšmė.

Dešinieji (jaunesnieji) atsilaisvinę bitai yra užpildomi nuliais.

Jeigu postūmyje per vieną bitą ženklo bitas nekeičia reikšmės, tai $OF = 0$, priešingu atveju, $OF = 1$.

Jeigu postūmis yra per kelis bitus, tada požymis OF yra neapibrėžtas.

SHR – SHift logical Right – loginis postūmis į dešinę.

1101 00vw mod 101 r/m [polinkis]

Jeigu $v = 0$, tai pastumiama į dešinę per vieną poziciją.

Jeigu $v = 1$, tai postūmio skaitliukas yra registro CL reikšmė.

Vyresnieji atsilaisvinę bitai yra užpildomi nuliais, kiekvienam išstumtam bitui.

Jeigu postūmyje per vieną bitą vyriausias bitas nekeičia reikšmės, tai $OF = 0$, priešingu atveju, $OF = 1$.

Jeigu postūmis yra per kelis bitus, tada požymis OF yra neapibrėžtas.

SAR – Shift Arithmetic Right – aritmetinis postūmis į dešinę.

1101 00vw mod 111 r/m [poslinkis]

Jeigu $v = 0$, tai pastumiama į dešinę per vieną poziciją.

Jeigu $v = 1$, tai postūmio skaitliukas yra registro CL reikšmė.

Vyresnieji atsilaisvinę bitai yra užpildomi nuliais arba vienetais, taip, kad būtų išsaugotas ženklas.

Jeigu postūmyje per vieną bitą ženklo bitas nekeičia reikšmės, tai $OF = 0$, priešingu atveju, $OF = 1$.

Jeigu postūmis yra per kelis bitus, tada požymis OF yra neapibrėžtas.

Pastaba. Postūmyje į dešinę loginis ir aritmetinis postūmis nėra tas pats, nes loginiame atsilaisvinę, atsiradę išstumtųjų vietoje, bitai yra užpildomi nuliais, o aritmetiniame postūmyje atsilaisvinę bitai yra užpildomi ženklo bitais.

ROL – ROtate Left – ciklinis postūmis į kairę.

1101 00vw mod 000 r/m [poslinkis]

Jeigu $v = 0$, tai pastumiama į dešinę per vieną poziciją.

Jeigu $v = 1$, tai postūmio skaitliukas yra registro CL reikšmė.

Stumiant per vieną bitą į kairę, vyriausias bitas užpildo jauniausio bito reikšmę. Jeigu postūmis yra daugiau nei per vieną poziciją, tai atitinkamai daugiau vyresniųjų bitų pernešama į jaunesnius.

Jeigu postūmyje per vieną bitą vyriausias bitas nekeičia reikšmės, tai $OF = 0$, priešingu atveju, $OF = 1$.

Vyriausias bitas užrašomas į požymį CF.

ROR – ROTate Right – ciklinis postūmis į dešinę.

1101 00vw mod 000 r/m [poslinkis]

Jeigu $v = 0$, tai pastumiama į dešinę per vieną poziciją.

Jeigu $v = 1$, tai postūmio skaitliukas yra registro CL reikšmė.

Stumiant per vieną bitą į dešinę, jauniausias bitas užpildo vyriausio bito reikšmę. Jeigu postūmis yra daugiau nei per vieną poziciją, tai atitinkamai daugiau jaunesniųjų bitų pernešama į vyresnius.

Jeigu postūmyje per vieną bitą vyriausias bitas nekeičia reikšmės, tai $OF = 0$, priešingu atveju, $OF = 1$.

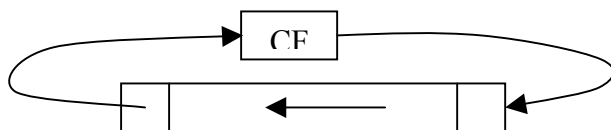
Jauniausias bitas užrašomas į požymį CF.

RCL – Rotate Left through Carry – perstumti cikliška į kairę, panaudojant pernešimo bitą CF.

1101 00vw mod 010 r/m [poslinkis]

Jeigu $v = 0$, tai pastumiama į dešinę per vieną poziciją.

Jeigu $v = 1$, tai postūmio skaitliukas yra registro CL reikšmė.

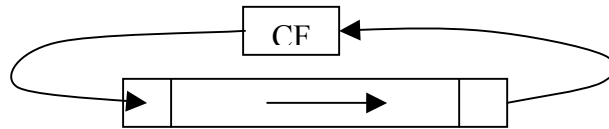


RCR – Rotate Right through Carry – perstumti cikliška į dešinę, panaudojant pernešimo bitą CF.

1101 00vw mod 011 r/m [poslinkis]

Jeigu $v = 0$, tai pastumiama į dešinę per vieną poziciją.

Jeigu $v = 1$, tai postūmio skaitliukas yra registro CL reikšmė.



AND – loginis “ir”.

Komanda naudojama formuoti nuliniams laukams.

1. registras / atmintis \wedge registras

0010 00dw mod reg r/m [poslinkis]

2. betarpiškas operandas \wedge registras / atmintis

1000 00sw mod 100 r/m [poslinkis] bet.op.j.b [bet.op.v.b.]

Jeigu s:w = 01, tada betarpiškas operandas yra imamas iš 16 komandos bitų.

Jeigu s:w = 11, tada betarpiškas operandas yra imamas iš 8 komandos bitų ir išplečiamas iki 16 bitų – žodžio, pagal ženklo plėtimo taisyklę.

3. betarpiškas operandas \wedge akumulatorius

0010 010w bet.op.j.b. [bet.op.v.b., jei w = 1]

OR – loginis “arba”.

Komanda naudojama formuoti vienetiniams laukams.

1. registras / atmintis \vee registras

0000 10dw mod reg r/m [poslinkis]

2. betarpiškas operandas \vee registras / atmintis

1000 00sw mod 001 r/m [poslinkis] bet.op.j.b [bet.op.v.b.]

Jeigu s:w = 01, tada betarpiškas operandas yra imamas iš 16 komandos bitų.

Jeigu s:w = 11, tada betarpiškas operandas yra imamas iš 8 komandos bitų ir išplečiamas iki 16 bitų – žodžio.

3. betarpiškas operandas \vee akumulatorius

0000 110w bet.op.j.b. [bet.op.v.b., jei w = 1]

XOR – eXclusive OR – išimtinis “arba”

Operacijoje su vienetiniu lauku, komanda operando bitus keičia į priešingus.

Operacijoje su nuliniu lauku, operando lauko bitai yra išsaugomi.

1. registras / atmintis \wedge registras

0011 00dw mod reg r/m [poslinkis]

2. betarpiškas operandas ^ registras / atmintis
1000 00sw mod 110 r/m [poslinkis] bet.op.j.b [bet.op.v.b.]

Jeigu s:w = 01, tada betarpiškas operandas yra imamas iš 16 komandos bitų.

Jeigu s:w = 11, tada betarpiškas operandas yra imamas iš 8 komandos bitų ir išplečiamas iki 16 bitų – žodžio.

3. betarpiškas operandas ^ akumulatorius
0011 010w bet.op.j.b. [bet.op.v.b., jei w = 1]

TEST – patikrinimas

Atliekama operacija, analogiška operacijai AND, tačiau rezultatas nėra išsaugomas, o formuojami požymių bitai:

1. CF = 0 ir OF = 0;
2. AF – neapibrėžtas;
3. PF, SF, ZF – jų reikšmės keičiamos pagal rezultato reikšmę.

Komanda yra naudojamas su valdymo prieš valdymo perdavimo komandas

1. registras / atmintis ~ registras
1000 010w mod reg r/m [poslinkis]

2. betarpiškas operandas ~ registras / atmintis
1111 011w mod 000 r/m [poslinkis] bet.op.j.b [bet.op.v.b., jei w = 1]

3. betarpiškas operandas ~ akumulatorius
1010 100w bet.op.j.b. [bet.op.v.b., jei w = 1]

Eilutinės komandos

REP – REPeate – pakartoti (pakartojimo prefiksas, priedas prie eilutinės komandos). Pakartojimo prefiksas sąlygoja sekančios komandos pakartojimą, jeigu ZF = 1 (t.y., yra nulis arba lygu, pagal situaciją) ir registre CX esanti reikšmė nelygi nuliui.

Sinonimai: **REP** / **REPZ** / **REPE** – REPeate / REPeate if Zero / REPeate if Equal – pakartoti / pakartoti, jei yra nulis / pakartoti, jei lygu.

1111 0011



z bitas, kuris yra lygus 1, ZF = z

Registro CX reikšmę keičia eilutinė komanda.

Komanda su pakartojimo prefiksu, kuris iššaukia ciklinį vykdymą: REP komanda .

REP komanda vykdymas gali būti pertrauktas. Pertraukimas įsimena tik vieną prefiksą ir jį atstato. Prefiksas REP turi būti betarpiškai prieš komandą, kuri yra

kartojama. Jeigu yra panaudoti papildomai vienas arba du prefiksai (segmento keitimo, magistralės blokavimo), tai po pertraukimo šie prefiksai nebeatstatomi, ir pratęsus vykdymą rezultatai gali būti nekorektiški. Todėl prieš tokį prefiksinį veiksmą su pakartojimu reikia uždrausti pertraukimus, o atlikus veiksmą vėl juos leisti. Kita vertus, gali gautis neleistinai ilgas laiko tarpas, kai pertraukimai neleidžiami.

Pastaba. Prefikso bitas *z* neturi įtakos eilutinėms komandoms MOVs, LODs, STOS. Neturi įtakos ir neeilutinėms komandoms, formaliai prieš jas pritaikius pakartojimo prefiksą, tačiau tos komandos nemažina registre CX esančios reikšmės, todėl tokia pora – pakartojimo prefiksas ir neeilutinė komanda, maža turi prasmės.

REPZ / REPNE – REPeate if Not Zero / REPeate if Not Equal – pakartoti, jei yra ne nulis / pakartoti, jei nelygu .

Pakartojimo prefiksas sąlygoja sekančios komandos pakartojimą, jeigu $ZF = 0$ (t.y., yra ne nulis arba nelygu, pagal situaciją) ir registre CX esanti reikšmė nelygi nuliui.

1111 0010

↑
z bitas, kuris yra lygus 1, $ZF = z$

Registro CX reikšmę keičia eilutinė komanda.

Pastaba. Pakartojimo prefikso bitas *z* neturi reikšmės komandoms MOVs, LODs, STOS, o tik toms eilutinėms komandoms, kuriomis atliekamas lyginimas arba skanavimas.

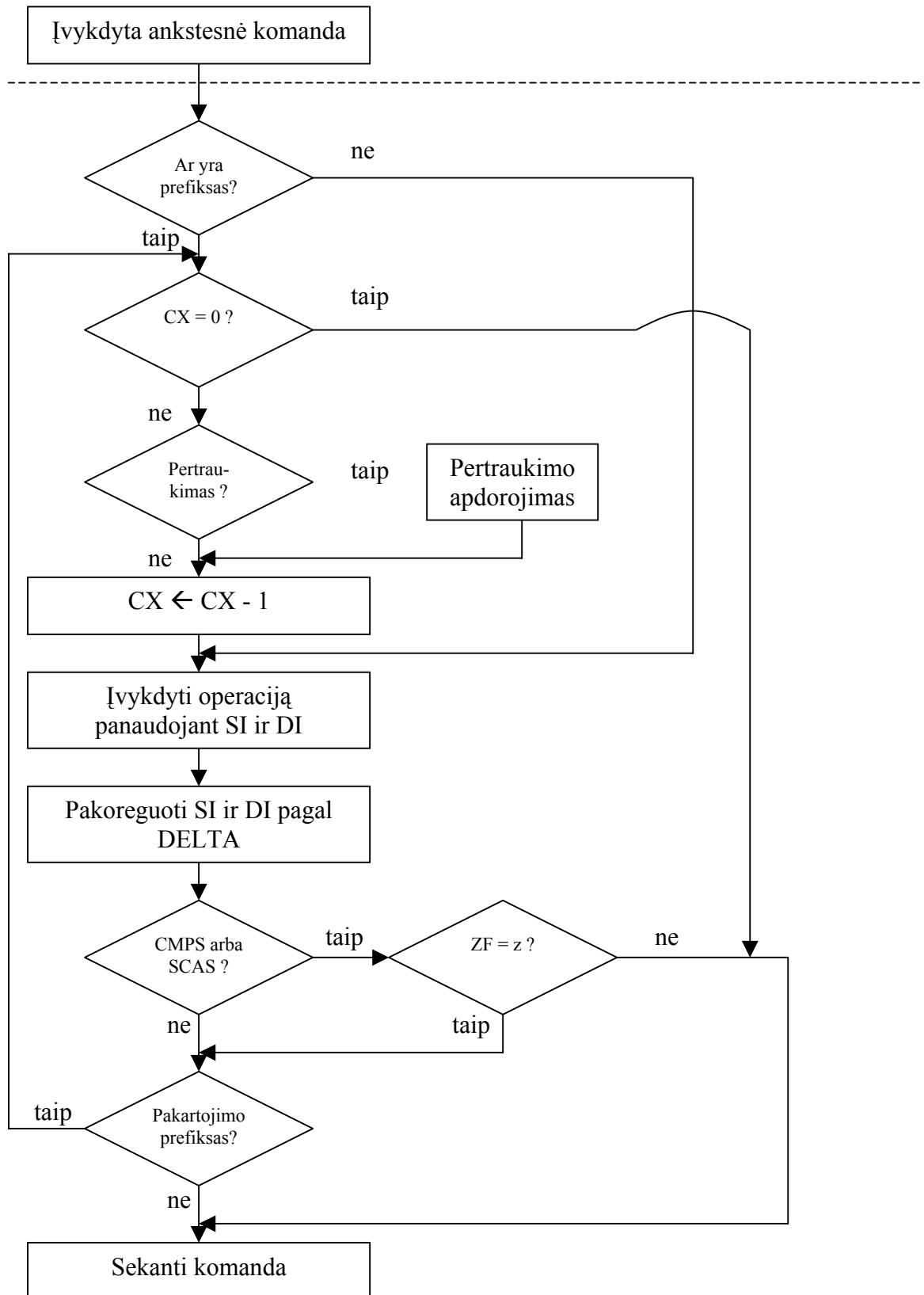
Eilutinės komandos su pakartojimo prefiksu įvykdymo schema

Pastaba. Veiksmą: $CX \leftarrow CX - 1$ atlieka eilutinė komanda, jeigu ji yra su pakartojimo prefiksu.

Pastaba. Pakartojimo prefiksas gali būti prieš bet kokią komandą.

	DF	DELTA
Baitas	0	+ 1
Baitas	1	- 1
Žodis	0	+ 2
Žodis	1	- 2

Prefiksas	z
REP / REPE / REPZ	1
REPNE / REPNZ	0



MOVS – MOVe String – eilutės persiuntimas.
1010 010w

Komanda yra persiunčiamas baitas arba žodis.

Jeigu $DF = 0$, tai indeksiniai registrai SI ir DI yra padidinami baitu (padidinama vienetu) arba žodžiu (padidinama dviem).

Jeigu $DF = 1$, tai indeksiniai registrai SI ir DI yra sumažinami baitu (sumažinama vienetu) arba žodžiu (sumažinama dviem).

Šaltinio lauko adresą nurodo registras SI, rezultato lauko adresą nurodo registras DI. Adresai yra segmento viduje (ribose).

Absoliutus rezultato lauko adresas suformuojamas naudojant registrus ES ir DI. Absoliutus šaltinio adresas gali būti formuojamas pasirinktinai naudojant registrus: CS, SS arba ES, ir DI.

CMPS – CoMPare String – eilučių palyginimas.
1010 011w

Vienu palyginimo veiksmu yra lyginami baitai arba žodžiai, iš šaltinio lauko atimant rezultato lauką. Suformuojami sąlygos požymiai, operandai nėra keičiami, tačiau modifikuojamos registrų SI ir DI reikšmės, pagal indeksinių registrų modifikavimo taisyklę – priklausomai nuo požymio DF (aprašyta prie komandos MOVS).

SCAS – SCAn String – simbolių paieška eilutėje.
1010 111w

Paieška yra atliekama palyginimu atėmimo būdu:

akumuliatorius - laukas, adresuojamas DI

Akumuliatorius – registras AL arba registras AX.

Laukas, adresuojamas DI – atitinkamai baito arba žodžio laukas, kurio adresas segmente yra nurodytas registre DI.

Atliekant komandą, atėmimo rezultate, yra suformuojami požymiai ir modifikuojamas registras DI, pagal indeksinių registrų modifikavimo taisyklę – priklausomai nuo požymio DF (aprašyta prie komandos MOVS).

.

LODS – LOaD String – eilutės užrašymas į akumuliatorių.
1010 110w

Baito laukas, nurodytas registru SI, užrašomas į akumuliatorių – registrą AL. Žodžio laukas, nurodytas registru SI, užrašomas į akumuliatorių – registrą AX.

Registras SI yra modifikuojamas pagal indeksinių registrų modifikavimo taisyklę – priklausomai nuo požymio DF (aprašyta prie komandos MOVS).

STOS – STOring String – registro įsiminimas eilutėje.

1010 101w

Priklausomai nuo to, ar bitas w yra 0 ar 1, registre AL arba AX esanti reikšmė yra įsiminama eilutėje, nurodytoje registru DI. Po to registras DI yra modifikuojamas, pagal indeksinių registrų modifikavimo taisyklę – priklausomai nuo požymio DF (aprašyta prie komandos MOVS).

Pastaba. Šiai komandai segmento registras visada yra ES, o nurodžius segmento keitimo prefiksą, jis yra ignoruojamas.

Valdymo perdavimo komandos

CALL – paprogramės (programos) iškvietimas

Komanda perduoda valdymą nurodytu adresu ir steke įsimena grįžimo adresą. Valdymo perdavimo adresas nurodomas operandu. Procedūrų iškvietimas skirstomas pagal segmentinę atminties organizaciją:

1. Iškvietimas segmento viduje (NEAR);
2. Iškvietimas kitame segmente (FAR).

Priklausomai nuo procedūros iškvietimo būdo, grįžimo adresas įsimenamas steke dvejopai: pirmu atveju kaip IP, antru – kaip IP : CS .

1. Vidinis tiesioginis (segmento viduje).

1110 1000 adr.j.b. adr.v.b.

Dviejų baitų adresu galima nurodyti bet kurią vietą segmente, bet komandoje nurodomas tiesioginis adresas atžvilgiu einamosios IP reikšmės.

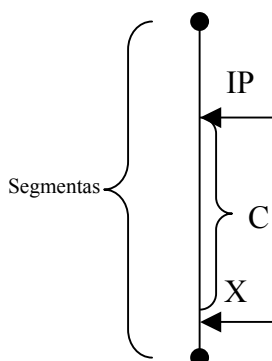
Veiksmai:

1. $SP \leftarrow SP - 2$.
2. *Steko viršūnė* $\leftarrow IP$.

Į steko viršūnę rašome IP reikšmę, taip išsaugomas grįžimo adresas – komandos, einančios po CALL, adresas.

3. $IP \leftarrow \text{tiesioginis adresas, nurodytas komandoje kaip operandas} + IP$.

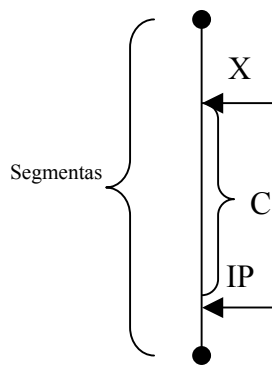
Tiesioginis adresas yra skaičius su ženklu žodyje, adresų diapazonas: -32768 – 32767. Valdymo perdavimo adresas apskaičiuojamas sudedant IP ir tiesioginį adresą.



X – vieta, kur norim perduoti valdymą

C – poslinkis

$X = IP + C$



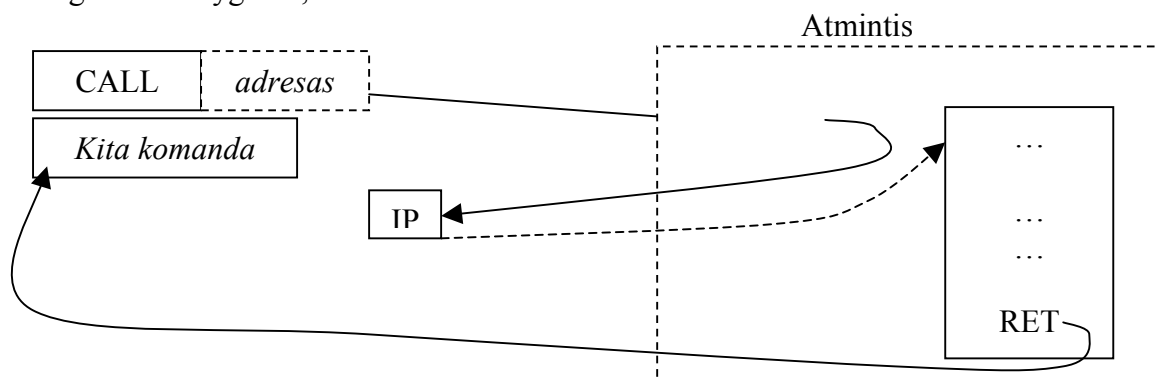
X – vieta, kur norim perduoti valdymą
 C – poslinkis
 $X = IP + (-C)$

2. Vidinis netiesioginis (segmento viduje)

1111 1111 mod 010 r/m posl.j.b. [posl.v.b.]

Valdymo perdavimo adreso reikšmė yra imama iš registro arba atminties, kuri yra nurodyta komandoje.

Jeigu mod nelygu 11, tada:



Pagal adresą, nurodytą komandoje, iš atminties lauko yra paimama reikšmė ir ji yra traktuojama kaip adresas, kuriuo reikia perduoti valdymą (adresas yra žodis).

Jeigu mod = 11, tada registre esanti reikšmė nusiunčiama į registrą IP, ir tokiu būdu ji yra traktuojama kaip adresas, kuriuo reikia perduoti valdymą.

3. Išorinis tiesioginis (už segmento ribų).

1001 1011 adr.j.b. adr.v.b. seg.reg.j.b. seg.reg.v.b.

Laukas seg.reg.j.b. seg.reg.v.b. yra nusiunčiamas į registrą CS.
 Laukas adr.j.b. adr.v.b. yra nusiunčiamas į registrą IP (tokiu būdu yra traktuojamas kaip poslinkis segmente).

Veiksmai:

1. $SP \leftarrow SP - 2.$
2. $Stekas \leftarrow CS.$
3. $SP \leftarrow SP - 2.$
4. $Stekas \leftarrow IP.$

5. $CS \leftarrow \text{segmento registro reikšmė, kuriame yra kviečiamoji procedūra.}$
6. $IP \leftarrow \text{poslinkio segmente reikšmė, kur yra kviečiamoji procedūra.}$

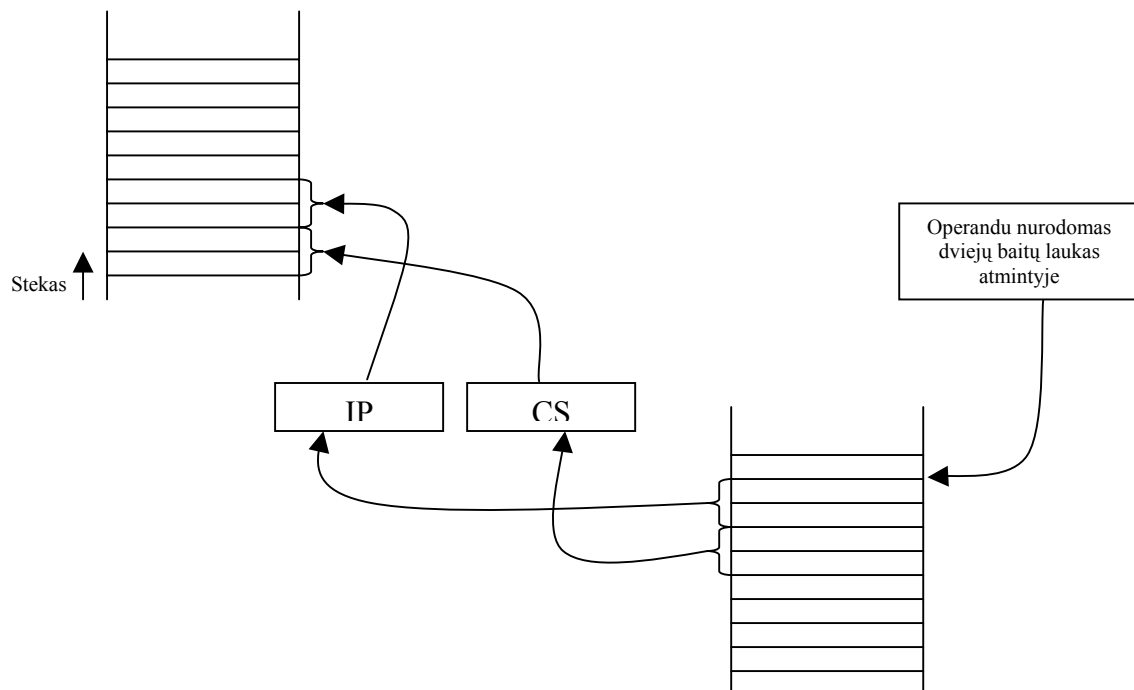
Kviečiama procedūra gali būti bet kurioje operatyvios atminties vietoje.

4. Išorinis netiesioginis.

1111 1111 mod 011 r/m [poslinkis]

Operandu nurodomas dviejų žodžių laukas atmintyje.

Komandos laukas mod negali būti lygus 11.



Vykdamą komandą, nurodyto lauko vyresnysis žodis nusiunčiamas į registrą CS, o jaunesnysis žodis nusiunčiamas į registrą IP. Taip yra atliekamas valdymo perdavimas. Į steką padedamos einamosios, prieš valdymo perdavimą buvusios, registrų CS ir IP reikšmės (einamoji IP registro reikšmė rodo į komandą, esančią po tuo metu vykdomos, šiuo atveju po CALL), todėl iš procedūros grįžtama teisingai.

RET – RETurn – grįžimas iš paprogramės.

Tai yra beadresinė komanda, yra skirtingi atvejai grįžimo segmento viduje ir išorėje.

1. Vidinė be steko išlyginimo.

1100 0011

Veiksmai:

1. $IP \leftarrow \text{Stekas.}$

2. $SP \leftarrow SP + 2.$

2. Vidinė su steko išlyginimu.

1100 0010 bet.op.j.b. bet.op.v.b.

Veiksmai:

1. $IP \leftarrow Stekas.$
2. $SP \leftarrow SP + 2.$
3. $SP \leftarrow SP + \text{betarpiškas operandas}.$

Pastaba. Betarpiškas operandas yra žodis.

3. Išorinis be steko išlyginimo.

1100 1011

Veiksmai:

1. $IP \leftarrow Stekas.$
2. $SP \leftarrow SP + 2.$
3. $CS \leftarrow Stekas.$
4. $SP \leftarrow SP + 2.$

4. Išorinis su steko išlyginimu.

1100 1010 bet.op.j.b. bet.op.v.b.

Veiksmai:

1. $IP \leftarrow Stekas.$
2. $SP \leftarrow SP + 2.$
3. $CS \leftarrow Stekas.$
4. $SP \leftarrow SP + 2.$
5. $SP \leftarrow SP + \text{betarpiškas operandas}.$

JMP – JuMP – besąlyginis perėjimas.

Kaip ir paprogramės iškviatimo komanda, JMP komandos skiriasi priklausomai nuo to, ar perėjimas yra segmento viduje ar išorėje. Skirtumas yra tas, kad nėra išsaugomos grįžimo adresas. Be to yra galimas valdymo perdavimas baito su ženklų ribose.

1. Vidinis artimas.

1110 1011 *poslinkis*

Poslinkis traktuojamas kaip skaičius su ženklu baite, diapazone: -128 – 127.

Skaičiuojant valdymo perdavimo adresą, baitas yra išplečiamas iki žodžio ir sudedamas su registre IP esančia reikšme.

2. Vidinis tiesioginis.
1110 1001 posl.j.b. posl.v.b.

Poslinkis traktuojamas kaip skaičius su ženklu žodyje, diapazone: -32768 – 32767.

Skaičiuojant valdymo perdavimo adresą, poslinkis yra sudedamas su registre IP esančia reikšme. Valdymo perdavimas atliekamas pridėdant arba atimant atitinkamą reikšmę, kurios diapazonas: -32768 – 32767, yra pakankamas perduoti valdymą į bet kurią vietą segmente.

3. Vidinis netiesioginis.
1111 1111 mod 100 r/m [poslinkis]

Tiesioginė adresacija komandoje JMP yra išreiškiama poslinkiu, o netiesioginėje adresacijoje valdymo perdavimo adresas yra imamas iš registro arba atminties žodžio ir nėra sudedamas su registre IP esančia reikšme.

Valdymo perdavimo adresas traktuojamas kaip skaičius be ženklo žodyje, todėl valdymo perdavimas galimas į bet kurią vietą segmente.

4. Išorinis tiesioginis.
1110 1010 adr.j.b. adr.v.b. seg.reg.j.b. seg.reg.v.b.

Laukas seg.reg.j.b. seg.reg.v.b. yra nusiunčiamas į registrą CS.
Laukas adr.j.b. adr.v.b. yra nusiunčiamas į registrą IP (toku būdu yra traktuojamas kaip poslinkis segmente).

Adresas čia yra traktuojamas kaip skaičius be ženklo ir jis yra siunčiamas į registrą IP, neatliekant sudėjimo su buvusiu registro IP reikšme, kaip vidinio tiesioginio valdymo perdavimo atveju.

5. Išorinis netiesioginis.
1111 1111 mod 101 r/m [poslinkis]

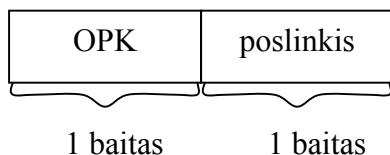
Operandu nurodomas dviejų baitų laukas atmintyje.

Komandos laukas *mod* negali būti lygus 11.

Vykdamas komandą, nurodyto lauko vyresnysis žodis nusiunčiamas į registrą CS, o jaunesnysis žodis nusiunčiamas į registrą IP.

Sąlyginis valdymo perdavimas

Yra 17 sąlyginio valdymo perdavimo komandų. Tų komandų formatai:



Vykdamą komandą yra tikrinami požymiai arba registro CX reikšmė ir, priklausomai nuo to tikrinimo rezultato, jei sąlyga yra patenkinta, tai prie registre IP esančios reikšmės yra pridedamas poslinkio baitas, kuris yra traktuojamas kaip skaičius su ženklu ir praplečiamas iki žodžio.

Komanda	Išplėstinis pavadinimas	Paaškinimas	Operacijos kodas	Tikrinama sąlyga
JA	Jump if Above	Pereiti, jeigu viršija	77	CF = 0 ir ZF = 0
JNBE	Jump if Not Below nor Equal	Pereiti, jeigu ne mažiau ir ne lygu		
JAE	Jump if Above or Equal	Pereiti, jei viršija arba lygu	73	CF = 0
JNB	Jump if Not Below	Pereiti, jei ne mažiau		
JNC	Jump if Not Carry	Pereiti, jeigu nėra pernešimo		
JB	Jump if Below	Pereiti, jeigu mažiau	72	CF = 1
JNAE	Jump if Not Above nor Equal	Pereiti, jeigu neviršija ir nelygu		
JC	Jump if Carry	Pereiti, jeigu pernešimas		
JBE	Jump if Below or Equal	Pereiti, jeigu mažiau arba lygu	76	CF = 1 arba ZF = 1
JNA	Jump if Not Above	Pereiti, jeigu neviršija		
JE	Jump if Equal	Pereiti, jeigu lygu	74	ZF = 1
JZ	Jump if Zero	Pereiti, jeigu nulis		
JCXZ	Jump if CX is Zero	Pereiti, jeigu CX = 1	E3	CX = 0
JG	Jump if Greater	Pereiti, jeigu daugiau	7F	ZF = 0 ir SF = OF
JNLE	Jump if Not Less nor Equal	Pereiti, jeigu ne mažiau ir nelygu		
JGE	Jump if Greater or Equal	Pereiti, jeigu daugiau arba lygu	7D	SF = OF
JNL	Jump if Not Less	Pereiti, jeigu ne mažiau		
JL	Jump if Less	Pereiti, jeigu mažiau	7C	SF nelygu OF

JNGE	Jump if Not Greater nor Equal	Pereiti, jeigu ne daugiau ir nelygu		
JLE	Jump if Less or Equal	Pereiti, jeigu mažiau arba lygu	7E	ZF = 1 arba SF nelygu OF
JNG	Jump if Not Greater	Pereiti, jeigu ne daugiau		
JNE	Jump if Not Equal	Pereiti, jeigu nelygu	75	ZF = 0
JNZ	Jump if Not Zero	Pereiti, jeigu ne nulis		
JNO	Jump if Not Overflow	Pereiti, jeigu ne perpildymas	71	OF = 0
JNP	Jump if Not Parity	Pereiti, jeigu nėra lyginumo	7B	PF = 0
JPO	Jump if Parity Odd	Pereiti, jeigu bitų suma nelyginė		
JP	Jump if Parity	Pereiti, jeigu yra lyginumas	7A	PF = 1
JPE	Jump if Parity Even	Pereiti, jeigu bitų suma lyginė		
JO	Jump if Overflow	Pereiti, jeigu perpildymas	70	OF = 1
JNS	Jump if No Sign	Pereiti, jeigu ženklo bitas yra nulis	79	SF = 0
JS	Jump if Sign	Pereiti, jeigu ženklo bitas yra vienetas	78	SF = 1

Ciklų komandos

LOOP – ciklas.

E2 poslinkis

Poslinkis yra vieno baido.

Pakartojimas vykdomas tiek kartų, kokia yra registro CX reikšmė.

Vykdamas yra mažinama registro CX reikšmė: $CX \leftarrow CX - 1$.

Jeigu registre CX esanti reikšmė nelygi nuliui, tai sekanti komanda nustatoma taip: *einamasis IP + poslinkis, paimtas iš komandos* : $IP \leftarrow IP + \text{poslinkis}$.

Jeigu registre CX esanti reikšmė lygi nuliui, tai vykdoma kita, toliau esanti, komanda.

Ciklo komanda valdymas gali perduodamas per -128 – 127 baitus.

Maksimalus ciklo pakartojimo skaičius yra 65 536.

LOOPE – LOOP if Equal – pakartoti jeigu lygu.

E1 *poslinkis*

Poslinkis yra vieno baido.

Pakartojimas vykdomas tiek kartų, kokia yra registro CX reikšmė.

Vykiant yra mažinama registro CX reikšmė: $CX \leftarrow CX - 1$.

Jeigu registre CX esanti reikšmė nelygi nuliui ir požymis $ZF = 1$, tai sekanti komanda nustatoma taip: *einamasis* $IP + poslinkis$, *paimtas iš komandos* : $IP \leftarrow IP + poslinkis$.

Jeigu registre CX esanti reikšmė lygi nuliui, tai vykdoma kita, toliau esanti, komanda.

Ciklo komanda valdymas gali perduodamas per -128 – 127 baitus.

Maksimalus ciklo pakartojimo skaičius yra 65 536.

Alternatyvi šios komandos mnemonika – **LOOPZ** – LOOP if Zero – pakartoti, jeigu nulis.

LOOPNE – LOOP if Not Equal – pakartoti jeigu nelygu.

E0 *poslinkis*

Poslinkis yra vieno baido.

Pakartojimas vykdomas tiek kartų, kokia yra registro CX reikšmė.

Vykiant yra mažinama registro CX reikšmė: $CX \leftarrow CX - 1$.

Jeigu registre CX esanti reikšmė nelygi nuliui ir požymis $ZF = 0$, tai sekanti komanda nustatoma taip: *einamasis* $IP + poslinkis$, *paimtas iš komandos* : $IP \leftarrow IP + poslinkis$.

Jeigu registre CX esanti reikšmė lygi nuliui, tai vykdoma kita, toliau esanti, komanda.

Ciklo komanda valdymas gali perduodamas per -128 – 127 baitus.

Maksimalus ciklo pakartojimo skaičius yra 65 536.

Alternatyvi šios komandos mnemonika – **LOOPNZ** – LOOP if Not Zero – pakartoti, jeigu ne nulis.

Pertraukimų komandos

INT – INTerrupt – programinis pertraukimas.

CD *tipas*

Atliekami veiksmi:

1. $SP \leftarrow SP - 2$
2. $Stekas \leftarrow SF$
3. $SP \leftarrow SP - 2$
4. $Stekas \leftarrow CS$
5. $SP \leftarrow SP - 2$

6. $Stekas \leftarrow IP$
7. $IF \leftarrow 0$
8. $TF \leftarrow 0$
9. $CS \leftarrow tipas \cdot 4 + 2$
10. $IP \leftarrow tipas \cdot 4$

9-tu ir 10-tu veiksmiais yra perduodamas valdymas į pertraukimo apdorojimo programą.

CD 3 – sustojimo taško pertraukimas yra ekvivalentus specialiai pertraukimo komandai iš vieno baido – CC. Komanda assembleriu užrašoma INT 3, o mašininiai kodai CD 3 ir CC yra ekvivalentūs:

$$CS \leftarrow 3 \cdot 4 + 2 = CC + 2$$

$$IP \leftarrow 3 \cdot 4 = CC$$

Komanda INT 4, kurios kodas yra CD 4, ekvivalenti komandai **INTO** – INTerrupt if Overflow – pertraukimas jeigu yra perpildymas. Pertraukimas įvyksta, jeigu OF = 1, ir :

$$CS \leftarrow 4 \cdot 4 + 2 = 10h + 2$$

$$IP \leftarrow 4 \cdot 4 = 10h$$

IRET – INTerrupt Return – grįžimas iš pertraukimo apdorojimo programos.
CF

Atliekami veiksmi:

1. $IP \leftarrow Stekas$
2. $SP \leftarrow SP + 2$
3. $CS \leftarrow Stekas$
4. $SP \leftarrow Sp + 2$
5. $SF \leftarrow Stekas$
6. $SP \leftarrow SP + 2$

Procesoriaus būsenos valdymas

Yra septynios komandos požymių CF, DF ir IF valdymui.

CLC – CLear Carry flag – išvalyti pernešimo požymį: $CF \leftarrow 0$

1111 1000

STC – SeT Carry flag – nustatyti pernešimo požymį: $CF \leftarrow 1$

1111 1001

CMC – CoMplement Carry flag – invertuoti požymį CF

1111 0101

CLD – CLear Direction flag – išvalyti krypties požymį: $DF \leftarrow 0$

1111 1100

STD – SeT Direction flag – nustatyti krypties požymį: $DF \leftarrow 1$

1111 1101

CLI – CLear Interrupt flag – išvalyti pertraukimo požymį: $IF \leftarrow 0$

1111 1010

STI – SeT Interrupt flag – nustatyti pertraukimo požymį: $IF \leftarrow 1$

1111 1011

Mikroprocesoriaus ir išorinių įvykių sinchronizavimo komandos

HLT – HaLT – sustabdyti.

1111 0100

Komanda perveda procesorių į sustojimo būseną, iš kurios procesorius gali išeiti šiais atvejais:

1. RESET signalas CLR.
2. Operatorinis pertraukimas NMI.
3. Išorinis pertraukimas, jeigu jis neuždraustas, t.y., $IF = 1$.

Komanda HLT yra perėjimas į pertraukimo laukimo būseną.

WAIT – laukti.

1001 1011

Komanda kas penki taktai tikrina signalą TEST. Sulaukęs signalo TEST, pradeda vykdyti kitą komandą, esančią po WAIT komandos.

WAIT komandos paskirtis yra procesoriaus pristabdymas, kol išorinis įrenginys baigs darbą. Jeigu $TEST = 1$, tada komanda WAIT yra nurodytas laukimas, o jeigu $TEST =$

0, tai vykdoma kita komanda. Signalą TEST valdo išoriniai įrenginiai, tame tarpe ir koprocesorius 8087. Mikroprocesorius laukimo metu gali sureaguoti į pertraukimus, kuriuos įvykdęs vėl grįžta į laukimo būseną.

LOCK – LOCK the bus – uždaryti magistralę.

1111 0000

Šitas prefiksas gali būti panaudotas prieš bet kokią komandą. Jis priverčia procesorių Intel 8088 blokuoti magistralę tos komandos vykdymo metu, kad joks kitas procesorius negalėtų naudoti magistralės.

Magistralė sujungia procesorių 8088 su išore: tai yra multikpleksinė adreso/duomenų magistralė. Bendra magistralė gali būti valdoma savu kontrolieriu. Procesorius Intel 8088 gali dirbti dviem režimais:

1. Valdančiosios magistralės signalus generuoja pats 8088 procesorius.
2. Magistralę valdo magistralės kontrolieris – speciali mikroschema.

Komanda LOCK monopolizuoja priėjimą prie atminties, tam kad netrukdytų kiti sistemos procesoriai.

ESC – ESCape – nubėgti.

1101 1xxx mod yyy r/m [poslinkis]

Šios komandos pagalba koprocesorius 8087 gauna savo komandas ir operandus procesoriaus 8088 darbo metu. Pats procesorius 8088 tuo metu nieko nedaro, tik iš atminties paima operandą ir jį nusiunčia į magistralę.

Procesoriaus 8088 komandos skaitymo iš atminties metu, koprocesorius 8087 nuolat stebi magistralę ir „pasigauna“ komandą ESC. Po to koprocesorius 8087 kontroliuoja magistralę ir „pasigauna“ adresavimo baitą. Bitai xxx ir yyy iš komandos ESC pirmų dviejų baitų traktuojami kaip koprocesoriaus operacijos kodas. Kuomet procesorius 8088 nusiunčia operando adresą į magistralę (jeigu komandos operandas buvo atmintis), koprocesorius tą operando adresą „pagauna“ ir pratęsia pradėtos koamndos vykdymą. Į procesoriaus Intel 8088 registrus tada niekas nepatenka. Jeigu laukas *mod* = 11, tai operandas yra registre, todėl procesorius 8088 nieko nedaro.

Pavyzdys.

DED9

1101 1110 1101 1001
ESC xxx mod yyy r / m

Procesorių 8088 ir 8087 registrai vienas kitam nėra prieinami, ir informaciją jie gali perduoti tik per atmintį. Jeigu laukas *mod* = 11, tada komanda ESC mikroprocesoriuje veikia kaip NOP. Komandos ESC prasminis operandas yra operandas atmintyje, o komandą su operandu registre mikroprocesorius įvykdo kaip NOP.

Ar operandas yra baitas ar žodis, priklauso nuo koprocesoriaus operacijos xxx yyy.

